IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 4, 2020, accepted August 17, 2020, date of publication August 31, 2020, date of current version September 14, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3020318

DPSO and Inverse Jacobian-Based Real-Time Inverse Kinematics With Trajectory Tracking Using Integral SMC for Teleoperation

HAMZA KHAN, SAAD JAMSHED ABBASI, AND MIN CHEOL LEE, (Member, IEEE)
School of Mechanical Engineering, Pusan National University, Busan 46241, South Korea

Corresponding author: Min Cheol Lee (mclee@pusan.ac.kr)

This work was supported in part by the nuclear research and development program through the National Research Foundation of Korea (NRF), funded by the Ministry of Science and ICT (MSIT, Korea) under Grant NRF-2019M2C9A1057807 and Technology Innovation Program (10073147, Development of Robot Manipulation Technology by Using Artificial Intelligence) funded By the Ministry of Trade, Industry & Energy (MOTIE, Korea).

ABSTRACT A six-degree-of-freedom robotic manipulator inverse kinematics (IK) for position control is proposed for the bilateral teleoperation process that is implemented through a joystick for nuclear power plant dismantling operations. The control strategy of the manipulator includes the use of the joystick to generate the Cartesian space trajectory followed by the IK to yield the joint space trajectory for implementing position control. In this paper, a novel technique for the IK is proposed. It involves the use of the particle swarm optimization (PSO) algorithm with the inverse Jacobian (IJ). The special case of the dual PSO is based on dividing the PSO algorithm into two such that the trajectory position and orientation are separately optimized by the algorithms, resulting in a faster convergence. In contrast, the inverse Jacobian aids in generating a smooth joint trajectory. The integral sliding mode control (ISMC) is proposed for position control because it does not require information on system dynamics. The ISMC improves the system trajectory tracking performance by using a switching gain to compensate for system dynamics and perturbations (disturbance and unmatched uncertainties), ultimately reducing the time delay. The effectiveness of the PSO combined with the IJ and the robustness of the ISMC in the teleoperation process are confirmed by the experimental results.

INDEX TERMS Teleoperation, inverse kinematics, robotic manipulator, PSO, inverse Jacobian, ISMC.

I. INTRODUCTION

The capability of robots to operate autonomously and their competence in performing numerous tasks have attracted the interest of researchers and industries [1]. The scope of industrial robot applications has been established from conventional handling, assembly, and welding tasks to a wide range of production activities [2]. Accordingly, robots and their precise operation have increasingly gained popularity. Moreover, their use to accomplish tasks that are demanding and dangerous to humans has significantly increased [3]. In industrial applications, the main objective of replacing humans with robots is to achieve efficiency and accuracy. For nuclear power generation, the utilization of human-less automation is required because of the hazardous thermal and radioactive environment of nuclear power plants (NPPs) [4]. The second stage after the shutdown of an NPP is dismantling, and recently, this phase has become a critical problem. After the NPP disaster in Fukushima, Japan, NPP dismantling has gained worldwide attention [5]. As a result, researchers and various industries have continued to introduce new robot technologies that are feasible for NPP dismantling in which robots are mainly used for cutting and cleaning [6]–[8]. For robotic control, the automotive or bilateral teleoperation process may be employed to perform crucial functions in such scenarios [9], [10].

To accomplish the desired tasks, the robots are guided by humans through end-effector trajectory planning (position and orientation), joint trajectory planning, joint motion control, and joint torque calculation and generation to accurately track trajectories [11]. For each given task, the direct control of the manipulator is not possible because there is no guidance or path to be tracked for the task. Accordingly, guidance starts with planning a trajectory for the manipulator's end-effector. This trajectory is then divided into several points. The inverse kinematics (IK) of the robotic manipulator aids

The associate editor coordinating the review of this manuscript and approving it for publication was Nishant Unnikrishnan.

the robot to find the set of joint angles (joint trajectory) for each point, aligning the robotic manipulator posture (position and orientation) with the desired posture. The next step is focused on the joint motion controller; each joint trajectory becomes the target set-point of each linked joint motion controller. Thereafter, to track the joint trajectory, sufficient joint torque has to be generated [12].

The IK solution has a critical function in robotics. The kinematics of robots describes the joint displacement relationship with the end-effector motion [13]. Usually, the end-effector position considered for the IK solution to introduce orientation causes complexity. The tasks assigned to the robot are in the Cartesian space, whereas the control is in the joint space. Therefore, through the IK, the joint trajectory is generated for the desired end-effector trajectory. Different techniques, including analytical and numerical methods, have been used for the IK of robotic manipulators. The analytical method, which consists of geometric and algebraic approaches [14], [15], is suitable for manipulators with less degree of freedoms (DOFs). This is because the solution becomes complex when the DOF is increased. The numerical method is an iterative method based on the inverse Jacobian, pseudo-inverse Jacobian, or Jacobian transform [16]–[18].

A. INVERSE KINEMATICS RELATED WORK

To date, different iterative methods have been used with the IK in various ways to solve robotic problems. The closed loop inverse kinematics (CLIK) algorithm, which implements the IK iteratively to obtain the desired pose (position and orientation) is proposed in [19]. The proposed algorithm is based on the pseudo-inverse of the Jacobian matrix, J. This approach was evaluated on a 6-DOF selective compliance articulated robot arm (SCARA), but the results were not satisfactory because the solution involved high computational complexity. A numerical method of IK based on the inverse Jacobian for a 5-DOF manipulator is proposed in [20]; however, it is concluded that when the inverse of the Jacobian is zero, singularity occurs. An iterative method for the IK using the pseudo-inverse Jacobian for a 7-DOF manipulator is presented in [16]. However, this Jacobian-based IK solution, which is an adapted method, requires calculation when both position and orientation are to be solved.

To overcome the computational complexity and extended calculation time, researchers have proposed many alternatives. Among these are the heuristic optimization techniques, which are emerging in robotics. Heuristic techniques include genetic algorithms [21], ant–bee colony [22], and particle swarm optimization (PSO) [23]. In these techniques, the solution is iteratively improved by adopting a set of operations that mimics a natural process, such as the flocking of birds. For a non-linear constrained problem, the most promising solution is the PSO. Despite the different proposed techniques for IK, constraints are barely considered. Many research works have implemented the IK using the PSO and evaluated their results for different types of robotic manipulators. In [23], the PSO was used to statistically analyze the IK in robotics; it was assessed in its capability of easily handling the IK of the robotic manipulator without computing the inverse model. In [24], the PSO was employed to solve the IK problem of a 4-DOF robotic manipulator based on the full resampling of particles. The PSO was used for an IK with high DOF and yielded the desired position and orientation with inconsiderable error; a self-collision avoidance was also added to the IK using the PSO [25]. Single-objective and multi-objective PSOs were proposed for 5-DOF and 7-DOF robotic manipulator end-effector positions in [26]. The idea of using the IK with the PSO for a cluttered environment with obstacles was presented in [27]. In this approach, the particles are separated into subgroups with specific tasks to achieve a faster 3-DOF planner robot convergence. In [28], the IK solution based on the PSO algorithm for a 7-DOF robot manipulator was proposed; the results were validated through simulation. An improved PSO (IPSO) for the solution of IK was presented in [29]; the algorithm was also simulated. A new PSO paradigm, known as dual particle swarm optimization (DPSO), which divides the PSO into two algorithms (PSO-1 and PSO-2), was introduced in [30]. The two algorithms are used to determine the IK for position and orientation separately. Accordingly, the solution converges faster toward the desired posture. After generating the joint trajectory using the IK, the second exigent task is the design and implementation of the manipulator motion control.

B. POSITION CONTROL RELATED WORK

Different control algorithms have been proposed and implemented in the robotics field to achieve reliable manipulator performance. Each controller has certain advantages and disadvantages. In the robotics industry, the proportional–integral–derivative (PID) controller, starting from the conventional PID control, has a long history [31]–[34]. It is a linear control and is extremely easy to implement; the problem is, it cannot control non-linear systems. Consequently, the sliding mode control (SMC), widely known as a robust non-linear control, was introduced [35]–[37].

The SMC is robust against parametric uncertainties and system non-linearities, but it has some drawbacks. This control consists of reaching and sliding phases in which the system reaches the sliding surface and remains on it. However, because of non-linearities, high switching gains are required, causing the system to shift back and forth on the desired surface and introduce chatter to the system response [38]. To remove the chatter, a non-linear compensator, known as sliding perturbation observer (SPO) [39], [40], is introduced. The SPO uses the partial feedback (position only) of the system to eliminate the effect on non-linearities from the system response by estimating the system state and perturbation (non-linearities, uncertainties, and disturbances); this estimated perturbation is useful in removing the chatter [41]. The integration of the SMC with the SPO (SMCSPO) produces a robust non-linear controller even in the presence of non-linearities [42].

The problem with using such a non-linear control is that it requires certain information regarding the system dynamic model (in the case of SPO, this is not accurate) to precisely control and estimate the perturbation. Moreover, it is exigent to derive a dynamic model for a 5-DOF robot. Consequently, another non-linear controller, known as integral sliding mode control (ISMC), has been introduced for the motion control of manipulators. The advantage of the ISMC is that it does not require system dynamics [43], [44]. The ISMC switching gains compensate for the absence of perturbation and system dynamics; hence, it has an edge over other controllers in terms of simplicity and robustness.

Another crucial problem is time delay, which is a phenomenon observed in various engineering systems [45]–[47]. Usually, these time delays are in communication (mostly in the teleoperation process), physical model, and control process. Currently, in the actual system, time delay is either inexistent or negligible. However, to observe the controller performance and stability in the presence of time delay, artificial communication and control process time-varying delays are introduced.

In the formulation of a practical control problem, the actual plant and its mathematical model used for designing the controller differ. This difference or mismatch emanates from unknown disturbances, plant parameters, and unmodeled dynamics, resulting in control process delays and system instability [48]–[50]. There are two phases in the sliding mode control: reaching phase (the plant reaches a predefined sliding surface in finite time) and sliding phase (the plant tends to remain on the sliding surface). In the sliding phase, the system persists to be invariant to parametric and nonparametric uncertainties, whereas in the reaching phase, the system is sensitive to disturbances; therefore, stability is not guaranteed [43] and may lead to the time delay in the system response. Accordingly, a new control technique, namely integral sliding mode control, introduces a new auxiliary sliding mode control law (or sliding surface) to compensate for the bounded disturbances and uncertainties. The ISMC eliminates the reaching phase of the SMC by immediately starting the sliding mode; it makes the system response insensitive to disturbances and aids in reducing the control process or system response delay [45], [49].

In this research, an application of the bilateral teleoperation process to control the manipulator using a joystick for the NPP dismantling is presented; a haptic feedback is also integrated to enable the operator to feel the environment. Initially, the inverse Jacobian method for the IK was implemented in the presence of joint and workspace constraints. In the present study, in the robotic manipulator, which is a SCARA-type robot, the error of the inverse Jacobian in the orientation is greater than the defined error range because of the kinematic structure of the robot. To reduce the error, the IK with the PSO is implemented.

The drawback of the classical PSO is that it is time consuming. This is because numerous particles are required to find an optimal solution within the defined error range when solving for both position and orientation simultaneously. Accordingly, the new PSO paradigm, the DPSO, is introduced. The DPSO uses two PSOs—one for position and another for orientation. In certain cases, the DPSO drives the solution to converge toward the desired solution faster than the PSO.

When the PSO-based IK is implemented for a planned trajectory, the generated joint trajectory has a wider angle variation (up to 3 rad) between two points in the path. This is because the PSO attempts to find the solution randomly without considering the previous solution, making it impossible to implement the position control using the joint trajectory. Considering the defined error range and variation in the joint trajectory, a novel IK technique for the trajectory is introduced: the DPSO integrated with the inverse Jacobian (DPSOIJ). The first (starting) point in the trajectory is solved using the DPSO; then, the inverse Jacobian solves for the rest of the trajectory. The resulting error is less than that of the inverse Jacobian with a smooth joint trajectory, making the DPSOIJ compatible for position control implementation in the manipulator. This is then followed by the ISMC design and implementation for the virtual manipulator teleoperation control using a master joystick while considering the time delay in the system. The simulation and experimental results of the IK using the DPSOIJ and position control using the ISMC effectively summarize the performance of the DPSOIJ and ISMC.

The main contributions of this research may be summarized as follows.

i. The proposed DPSOIJ-based IK is a novel technique that generates a smooth trajectory with less errors in the desired posture.
ii. The design and implementation of a stable ISMC to compensate for the effect of disturbances and time delay controls the system without considering the system dynamics in teleoperation.

The rest of manuscript is organized as follows. Section 2 describes the system design and kinematics of the virtual manipulator for the NPP dismantling. The proposed IK algorithm is presented in Section 3. Section 4 discusses the mathematical design of the control algorithm. Section 5 explains the experimental setup, and Section 6 presents the simulations results. Section 7 elaborates on the experimental results, and Section 8 presents the concluding remarks.

II. SYSTEM DESIGN AND KINEMATICS MODEL

The robotic manipulator is a 6-DOF SCARA-type robot. The first link, shown in Figure 1, is composed of a telescopic mast, which is actually a prismatic joint. The other manipulator is a 5-DOF robotic arm; it is used for cutting applications during dismantling with a working tool at the end-effector. The computer-aided design (CAD) model of the SCARA is presented in Figure 1.

SolidWorks is employed to generate the virtual prototype of the actual system. The axis configuration is shown in Figure 2. The red lines indicate the axis of rotation

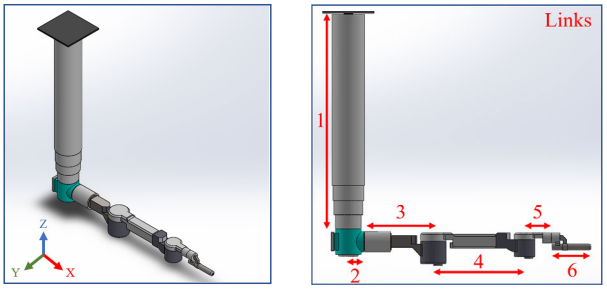

FIGURE 1. CAD model of 6-DOF SCARA type robot.

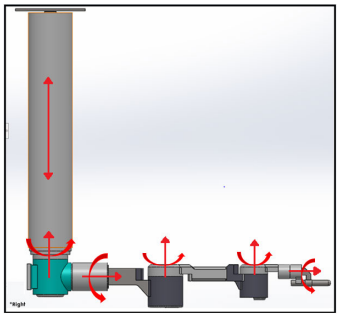

FIGURE 2. Axis configuration of 6-DOF SCARA robot.

TABLE 1. DH parameters of SCARA robot.

Joint	θ (rad)	d (mm)	a (mm)	α (rad)
1	0	d_1*	0	0
2	θ_2^*	0	0	$-\pi/2$
3	θ_3^*	750	0	$\pi/2$
4	θ_4^*	0	820	0
5	θ_5^*	0	0	$\pi/2$
6	θ_6^*	730	0	0

TABLE 2. Joint space limitations.

d_1^*	θ_2^*	θ_3^*	θ_4^*	θ_5^*	θ_6^*
2500∼10000	$-2\pi\sim2\pi$	$-2\pi\sim2\pi$	$\frac{-3\pi}{4}\sim\frac{3\pi}{4}$	$\frac{-3\pi}{4}\sim\frac{3\pi}{4}$	$-2\pi\sim2\pi$

TABLE 3. Joint information.

Joint	Type	Mass (kg)
1	Prismatic	345.9
2	Revolute	88
3	Revolute	139.7
4	Revolute	129.4
5	Revolute	20.8
6	Revolute	11.7

(single-headed arrow) and the prismatic joint of the 6-DOF SCARA robot. For the kinematics, the Denevit–Hartenberg (DH) parameters of the robot are summarized in Table 1. The robot joint space limitations are listed in Table 2, and the joint information on the manipulator is summarized in Table 3.

III. INVERSE KINEMATICS (IK)

In this section, the inverse kinematics of the 5-DOF robotic manipulator is discussed. The prismatic joint is considered and separately controlled from the robotic arm.

A. INVERSE JACOBIAN

The inverse Jacobian method is an iterative numerical method for solving the IK. The Jacobian matrix is a first-order partial derivative matrix of all the links [51]. The iterative form of the Jacobian method can be written as

$$\theta_{i+1} = \theta_i + d\theta = \theta_i + J^{-1}dP \tag{1}$$

where i is the number of iterations to be performed; $P = \left[P_x P_y P_z \emptyset\theta\varphi\right]^T$ is the posture matrix containing three position and three orientation coordinates; J^{-1} represents the inverse of the Jacobian matrix. Considering the 5-DOF robotic arm case, the coordinates for the general forward kinematics can be expressed as

$$\begin{bmatrix} dx \\ dy \\ dz \\ d\theta_x \\ d\theta_y \\ d\theta_z \end{bmatrix} = [J] \begin{bmatrix} d\theta_1 \\ d\theta_2 \\ d\theta_3 \\ d\theta_4 \\ d\theta_5 \end{bmatrix} \tag{2}$$

where dx, dy, and dz are the differential motions of the robot hand in the x, y, and z coordinate axes, respectively; $d\theta_x$, $d\theta_y$, and $d\theta_z$ are the changes in the end-effector roll, pitch, and yaw, respectively. Equation (2) can also be represented as

$$[D] = [J]\,[D_\theta] \tag{3}$$

where J is a 6×5 Jacobian matrix. To calculate angle θ, the IK can be written as

$$[D_\theta] = [J^{-1}]\,[D] \tag{4}$$

$$\begin{bmatrix} d\theta_1 \\ d\theta_2 \\ d\theta_3 \\ d\theta_4 \\ d\theta_5 \end{bmatrix} = [J^{-1}] \begin{bmatrix} dx \\ dy \\ dz \\ d\theta_x \\ d\theta_y \\ d\theta_z \end{bmatrix} \tag{5}$$

where J^{-1} should be calculated using J, which is not a square matrix. Therefore, to solve the inverse Jacobian matrix, a pseudo-concept is required [52]:

$$(JJ^T)\left(JJ^T\right)^{-1} = \mathrm{I} \tag{6}$$

$$J_{6\times5}\left[J_{5\times6}^T\left(J_{6\times5}J_{5\times6}^T\right)^{-1}\right] = JJ_{5\times6}^{-1} = \mathrm{I} \tag{7}$$

where $J_{5\times6}^{-1}$ is the new inverse Jacobian matrix, which is obtained using the pseudo-inverse concept [53]. The pseudo-inverse is given as

$$J^{-1} = J^T(JJ^T)^{-1} \tag{8}$$

where

$$J_{5\times6}^{-1} = J_{5\times6}^{T}(J_{6\times5}.J_{5\times6}^{T}) \tag{9}$$

Finally,

$$d\theta_{5\times1} = J_{5\times6}^{-1} dP_{6\times1} \tag{10}$$

where the Jacobian matrix at this point is expressed as (11) and (12), shown at the bottom of the next page.

where o_i is the distance between the origin and ith-axis coordinate; R_i^j is the orientation matrix from the ith-axis to jth-axis.

B. PARTICLE SWARM OPTIMIZATION (PSO)

The PSO is an optimizing technique that attempts to improve its population solution iteratively; its development is presented in [54]. In the original formulation, the authors presented an optimization technique that contains several particles (called swarm) attempting to optimize the problem by exploring in the problem search space. The measure of quality is a *fitness function (objective function), f(obj),* which depends on different parameters. The algorithm consists of several particles, and each particle position is considered as a possible solution to the problem. The particle iteratively changes its position with velocity. After the kth iteration, the position and velocity of the independent particle in the generated population, $\mathcal{P}$, at the kth iteration is defined by M-dimensional vectors as follows:

$$x^i(k) = \left[x_1^i(k) \ldots x_M^i(k)\right]^T \tag{13}$$

$$v^i(k) = \left[v_1^i(k) \ldots v_M^i(k)\right]^T \tag{14}$$

where $i = 1 \ldots \|\mathcal{P}\|$; x and v represent the position and velocity of the ith member of the M-dimensional vector, respectively. In the PSO, each particle also has information regarding its neighbors through information exchange; accordingly, the particles are aware of the globally best particle in the swarm, and the swarm follows the best particle. The communication of each particle with its neighbor is defined as

$$N_i(k) = \{j \in \mathcal{P} : i \leftrightarrow j\} \tag{15}$$

where the symbol $\leftrightarrow$ represents the information exchange between two particles. The movement and direction of particles within the search space in the PSO depends on the personal best solution, $X_{pB}^i(k)$, and the global best, $X_{gB}(k)$ [55]. When the neighbors communicate, they share the information regarding the best-known particle, which is globally best in the swarm. Based on this information, the new globally best particle is updated by satisfying the condition in Eq. (16):

$$X_{gB}(k) = \left\{X_{pB}^i(k) : f(obj) = X_{pB}^i(k) < X_{gB}(k)\right\} \tag{16}$$

where $X_{gB}(k)$, $X_{pB}^i(k)$, and $f\left(X_{pB}^i(k)\right)$ are the global best at the kth iteration, the personal best of the ith particle at the kth iteration, and the objective function to compute the personal best of the ith particle at the kth iteration, respectively. The particles at every iteration update their position and compare their personal best with the global best. If the personal best of a particle at the kth iteration in minimizing the problem is less than the global best described in Eq. (16), the global best is updated with that particle's personal best. The personal best of each particle is updated according to Eq. (17), i.e., if the personal best of the ith particle at the kth iteration is less than the last known personal best of the same particle, then X_{pB}^i updates as follows.

$$X_{pB}^i(k) = \left\{X_{pB}^i(k) : f(obj) < X_{pB}^i\right\} \tag{17}$$

As the PSO algorithm starts, the initial population with n number of particles is generated within the predefined range with nil velocity. At each iteration, the ith particle velocity and position are updated according to Eqs. (18) and (19):

$$v^i(k+1) = \omega.v^i(k) + c_1.rand.\left(X_{pB}^i(k) - x^i(k)\right) + c_2.rand.\left(X_{gB}(k) - x^i(k)\right) \tag{18}$$

$$x^i(k+1) = x^i(k) + v^i(k+1) \tag{19}$$

where ω represents the inertia coefficient; c_1 and c_2 are the learning or acceleration factors; *rand* represents the generation of random numbers within the specified range $[-v_{max}, v_{max}]$. As the particles approach the desired result, the velocity is optimized in such way that the process is decelerated without causing the solution to deviate from the target result or have a large or increasing error.

C. PROPOSED ALGORITHM

The analysis of the various techniques for solving the real-time IK of a 5-DOF robot for NPP dismantling includes the calculation of the end-effector position (in Cartesian space) and orientation (roll, pitch, and yaw) simultaneously. However, the allowable calculation time for the first point in the trajectory is considerably brief, i.e., $t_{allow} = 0.25s$. The maximum allowable tolerances for the position and orientation of the end-effector are $E_{Tp} = \pm 5mm$ and $E_{To} = \pm .0175$rad, respectively.

$$J = \begin{bmatrix} z_0 \times (o_5 - o_0) & z_1 \times (o_5 - o_1) & \cdots & z_4 \times (o_5 - o_4) \\ z_0 & z_1 & \cdots & z_4 \end{bmatrix} \tag{11}$$

$$z_0 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, z_1 = R_0^1 z_0, \cdots, z_4 = R_3^4 z_3 = R_0^1 \cdots R_3^4 z_0 \tag{12}$$

To achieve better results for a single point in Cartesian space, a new PSO paradigm, known as DPSO, was proposed to solve the IK problem using the combination of two different PSO algorithms. For the trajectory, a new combination of the inverse Jacobian and DPSO, known as dual particle swarm optimization integrated with the inverse Jacobian (DSPOIJ), was formulated.

1) DPSO

The PSO algorithm iteratively improves the solution by constantly updating the information in each particle based on the best solution thus far. Each particle moves with a certain velocity toward the desired point. As the particle approaches the best particle in the population, its velocity decreases. The proposed scheme uses two different PSO algorithms working separately—one for analyzing the position and the other for the orientation. First, PSO-1 attempts to reach the target position of the end-effector. Then, PSO-2 slightly adjusts the joint angles to obtain the desired orientation. The following steps describe the DPSO process.

Step 1: The initial population of N particles is randomly generated within the predefined range; N is the number of particles generated depending on the robot's DOF. It is defined as

$$N = \frac{n_{dof} \times d_N}{t_{allow} \times E_{Tp}} \tag{20}$$

where n_{dof} represents the robot's number of DOFs; d_N is a constant number ($d_N = 10$ in the simulation) for increasing the population range.

Step 2: The global best is set to infinity, and the forward kinematics is computed for each particle. The function of the forward kinematics of a robot whose joint configuration is defined as $q_i(k)$ is given by

$$f_{kine}(q_i(k)) = {}^0T_M(q_i(k)) = \begin{bmatrix} {}^0R^i_M(k) & {}^0p^i_M(k) \\ \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} & 1 \end{bmatrix} \tag{21}$$

where ${}^0R^i_M(k) \in \mathbb{R}^{3\times3}$ and ${}^0p^i_M(k) \in \mathbb{R}^{3\times1}$ represent the end-effector orientation and position, respectively.

Step 3: The objective function for each particle is computed.

$$\begin{aligned} f_{obj1} &= \|P_d - P_c\| \\ &= \sqrt{(x_d - x_c)^2 + (y_d - y_c)^2 + (z_d - z_c)^2} \end{aligned} \tag{22}$$

where $P_d\ (x_d, y_d, z_d)$ and $P_c(x_c, y_c, z_c)$ are the desired and current positions in the Cartesian space, respectively.

Step 4: Each particle's personal best is computed according to Eq. (17). Then the global best is selected according to Eq. (16). The position for the next particle is updated using Eqs. (18) and (19).

Step 5: The personal best and global best are computed at each iteration. Check whether the termination criteria have been satisfied. The algorithm can determine whether the desired joint variable for the required position has been obtained.

Step 6: If the required position is obtained within $E_{Tp} = \pm5mm$, terminate all iterations for the position.

Step 7: Generate M particles around the global best obtained in Step 6; M is defined as

$$M = \frac{it_{pso1} \times t_{pso1}}{E_{To} \times d_M} \tag{23}$$

where it_{pso1}, t_{pso1}, and d_M represent the total iterations performed by PSO-1 to obtain the desired position, the time used by PSO-1, and the constant that causes M < N, respectively. In this case $d_M = 10$. For PSO-2, the global best particle of PSO-1 with a random orientation of ±0.174 rad is used. The probability of maintaining the end-effector position obtained by PSO-1 while solving for the orientation in PSO-2 is greater by using the global best particle obtained by PSO-1 in PSO-2.

Step 8: The newly generated particles for PSO-2 start adjusting the joint variables in such a way that the end-effector position remains the same; however, the desired orientation is obtained by repeating Steps 2–5.

The particle velocity in PSO-2 is reduced as the newly generated particles move into the search space near the desired position. If the acceleration factor is high, then an output error will occur because the particle will search in the out-of-range space. To solve this problem, the new velocity used for PSO-2 is defined as follows.

$$v^i(k+1) = \omega.v^i(k) + c_2.rand.\left(X_{gB}(k) - x^i(k)\right) \tag{24}$$

The objective function of PSO-2 is

$$\begin{aligned} f_{obj2} &= \rho_{pso}\|P_d - P_c\| + \sigma_{pso}\|O_d - O_c\| \\ &= \rho_{pso}f_{obj1} + \sigma_{pso}\sqrt{(R_d - R_c)^2 + (P_d - P_c)^2 + (Y_d - Y_c)^2} \end{aligned} \tag{25}$$

where $O_d\ (R_d, P_d, Y_d)$ and $O_c(R_c, P_c, Y_c)$ are the desired and current orientations, respectively; R, P, and Y represent the roll, pitch, and yaw, respectively. Moreover, $0 \le \rho_{pso} \le 1$ and $0 \le \sigma_{pso} \le 1$ are the weighting importance of position and orientation, respectively. In the proposed algorithm, ρ_{pso} and σ_{pso} are set as 0.25 and 0.75, respectively, because PSO-2 searches for the desired orientation without affecting the position. However, in some cases, $\rho_{pso} = \sigma_{pso} = 1$, resulting in a single PSO or traditional PSO because of the complex desired posture. In such cases, only Eq. (25) is used for the entire solution of IK. Thus, it can be concluded that occasionally, either the traditional PSO or DPSO is employed. The current study generally focuses on the DPSO.

Step 9: Terminate the iterations if the desired orientation is obtained within $E_{Tp} = \pm5mm$ and $E_{To} = \pm.0175$ rad.

Step 10: The algorithm checks whether the results are within the space limitation of the robot. Otherwise, the algorithm will adjust them within the specified range.

The DPSO steps are summarized in the flowchart shown in Figure 3.

2) DPSO INTEGRATION WITH INVERSE JACOBIAN

After the separate implementation of the inverse Jacobian and DPSO methods, the selection was based on the errors

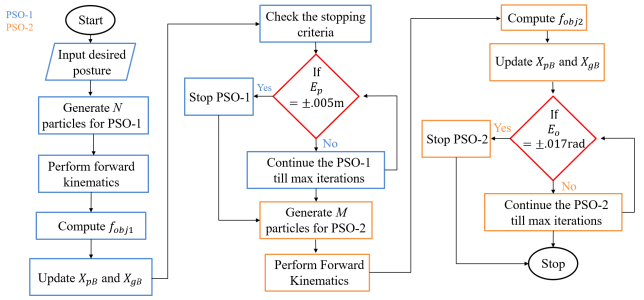

FIGURE 3. DPSO flowchart.

of position and orientation. For a single point, the inverse Jacobian has an orientation error exceeding the specified tolerance when the position and orientation are solved simultaneously. In contrast, the DPSO satisfies the error conditions. Subsequently, the desired trajectory for the end-effector is introduced and divided into several points; for each point, the IK was performed using both the above-mentioned methods.

The DPSO was ultimately found to be the worst method for the end-effector joint trajectory. There was a substantial variation in the joint angle for which it was impossible to implement position control for the robotic manipulator. In contrast, the joint trajectory was observed to be smooth using the inverse Jacobian. For a trajectory to complete the task with a smooth joint path and less errors, a new technique combining both the inverse Jacobian and DPSO is proposed. The mathematical expressions of both methods are the same as those given previously; the only aspect that has changed, however, is the Jacobian matrix. After observing the DPSOIJ responses, the first three rows of Eq. (11) are considered as forming the new Jacobian matrix. It is actually the position matrix given as follows.

$$J = \begin{bmatrix} z_0 \times (o_5 - o_0) & z_1 \times (o_5 - o_1) & \cdots & z_4 \times (o_5 - o_4) \end{bmatrix} \tag{26}$$

The proposed DPSOIJ algorithm works in a way similar to that in obtaining the design trajectory, which is divided into several points. The IK for the first point, which is the starting (initial) posture of the robotic manipulator, is performed using the DPSO. For the rest of the trajectory, the inverse Jacobian is employed. Based on the different simulations of IK using the inverse Jacobian, it is observed that the initial point in the trajectory must be accurately solved. When a robot has an initial orientation error caused by singularity, the rest of the trajectory orientation will have the same initial error in the subsequent desired trajectory even if the IK is used to reach the desired final orientation. Therefore, considering the behavior of inverse Jacobian, the initial IK point was solved using the DPSO to achieve accurate calculation and then followed by the inverse Jacobian to obtain a smooth joint trajectory.

IV. INTEGRAL SLIDING MODE CONTROL (ISMC)

In the field of robotics, various control algorithms, including linear and non-linear controllers, are widely used. In the non-linear controller type, the SMC technique is extremely useful for designing the controllers in the system with matched disturbance uncertainties. The compensated dynamics of the system become insensitive to disturbances and uncertainties in the case of sliding mode control. This insensitivity is only achieved by reaching the sliding surface and establishing sliding motion [49]. However, the SMC requires system dynamics to compensate for the non-linearities. Therefore, to overcome this problem in system dynamics, the ISMC is proposed. The ISMC actually compensates for the disturbance by designing an auxiliary sliding surface while retaining the order of the uncompensated system.

A. ISMC FORMULATION

The auxiliary sliding surface is defined as

$$s = \sigma - z \tag{27}$$

where s, σ, and z are the auxiliary sliding surface, conventional SMC sliding surface, and integral term, respectively. The actual sliding surface of SMC is given as

$$\sigma = \dot{e} + c \cdot e \tag{28}$$

where e and c are the error and positive constant, respectively. Consider an uncertain system given in the following state space equation:

$$\dot{x} = f(x) + B(x)u + \Psi(x, t) \tag{29}$$

where $x \in \mathbb{R}^n$ represents the state vector; $u \in \mathbb{R}^m$ represents the control input; $\Psi(x, t)$ represents the perturbation. For such systems, the dynamic form of the actual sliding surface is as follows.

$$\dot{\sigma} = \Psi(x, t) - u \tag{30}$$

It is assumed that $\Psi(x, t) < \mathrm{H}$, where H is the upper bound of perturbation. To verify the Lyapunov stability in the SMC case, the following condition must be satisfied.

$$\sigma \cdot \dot{\sigma} \leq 0 \tag{31}$$

Moreover, the control input should be greater than the upper bound of perturbation. In the ISMC, the control is divided into two parts as given by

$$u = u_1 + u_2 \tag{32}$$

where u_1 initially compensates for the bounded perturbation without reaching the phase, and u_2 causes the sliding variable to be equal to zero over time. The integral term of the ISMC is defined as follows.

$$\dot{z} = -u_2 \tag{33}$$

The auxiliary sliding variable dynamics, $\dot{s}$, can be calculated as

$$\dot{s} = \dot{\sigma} - \dot{z} \tag{34}$$

where $\dot{s}$ is obtained by integrating Eqs. (30), (32), and (34).

$$\dot{s} = \Psi(x,t) - (u_1 + u_2) - (-u_2) \tag{35}$$

After solving Eq. (35), $\dot{s}$, is obtained as follows.

$$\dot{s} = \Psi(x,\dot{x},t) - u_1 \tag{36}$$

In finite time, the control input, u_1, drives the auxiliary sliding variable to zero and is given as

$$u_1 = K \cdot sat(s) \tag{37}$$

where K represents the switching gain. In the auxiliary sliding mode, the equivalent control input can be calculated using the condition $s = 0$ ($\dot{s} = \mathbf{0}$).

$$\dot{s} = \Psi(x,\dot{x},t) - u_{1eq} = 0 \tag{38}$$

$$u_{1eq} = \Psi(x,\dot{x},t) \tag{39}$$

The actual dynamics of the system calculated using Eqs. (30), (32), and (39) during the auxiliary sliding mode are given as

$$\dot{\sigma} = \Psi(x,\dot{x},t) - (u_{1eq} + u_2) \tag{40}$$

$$\dot{\sigma} = \Psi(x,\dot{x},t) - \Psi(x,\dot{x},t) - u_2 \tag{41}$$

$$\dot{\sigma} = -u_2 \tag{42}$$

$$u_2 = k \cdot \sigma, k > 0 \tag{43}$$

where k is a constant. In the ISMC, the actual sliding surface starts and remains at zero because there is no reaching phase. To start the auxiliary sliding surface from zero without the reaching phase, $s = 0$ must be imposed in Eq. (27) such that the following are obtained.

$$s_{(0)} = \sigma_{(0)} - z_{(0)} = 0 \tag{44}$$

$$\sigma_{(0)} = z_{(0)} \tag{45}$$

$$z_{(0)} = \dot{x}_{d(0)} + c \cdot x_{c(0)} - x_{2(0)} - c \cdot x_{(0)} \tag{46}$$

As previously discussed, the ISMC does not require system dynamics, and only the feedback of the system output is required. The control input ISMC is then derived as follows.

$$u = \rho \cdot sat(s) + \mathrm{k} \cdot \sigma \tag{47}$$

B. STABILITY

The Lyapunov stability analysis is the most popular approach to prove the stability and evaluate the stable convergence property of the non-linear controller [56]. To investigate the ISMC stability, the Lyapunov function [57], [58] is defined as

$$V(t) = \frac{1}{2}s^2(t) \tag{48}$$

where $V(0) = 0$ and $V(t) > 0$ for $s(t) \neq 0$. Taking the first derivative of Eq. (48) with respect to time yields

$$\dot{V}(t) = s(t)\dot{s}(t) < 0, s(t) \neq 0 \tag{49}$$

where $\dot{(t)}$ is given as

$$\dot{(t)} = -Ksat(s) + \Psi(x,\dot{x},t) \tag{50}$$

in which K is the switching gain with the assumption that $K > \left|\Psi(x,x',t)\right|$. Substituting Eq. (50) into Eq. (49) yields the following.

$$\begin{aligned}\dot{V}(t) &= s(t)\left[-Ksat(s) + \Psi(x,\dot{x},t)\right] \\ &\leq -|s(t)|K + |s(t)|\Psi(x,\dot{x},t) \\ &\leq -|s(t)|\left[K - \left|\Psi(\mathrm{x},\mathrm{x}',\mathrm{t})\right|\right] < 0\end{aligned} \tag{51}$$

To satisfy Eq. (51), K should exceed the upper bound of the absolute value of perturbation, i.e., $K > \left|\Psi(x,x',t)\right|_{max}$. The foregoing analysis ensures stability because $\dot{V}(t)$ is negative.

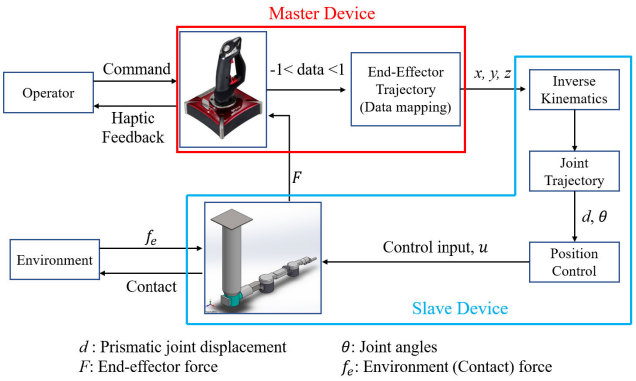

FIGURE 4. Bilateral teleoperation experimental structure.

V. EXPERIMENTAL ENVIRONMENT

The experimental structure of bilateral teleoperation is described in Figure 4. MATLAB/Simulink is used for the design, implementation, and validation of the proposed IK and position control algorithm.

A. VIRTUAL ROBOT MODEL

The modeling of robotic manipulators can be performed either by using the Euler–Lagrange method in terms of the equation of motion or by using a CAD model with the SimMechanics MATLAB toolbox to represent an actual system [59], [60]. SimMechanics is a block diagram environment for modeling and simulating mechanical systems that use the standard Newtonian dynamics of forces and torques [61]. The prototype of the actual system is modeled using SolidWorks and then exported to the Simulink environment. This is accomplished using the SimMechanics toolbox, which computes the system dynamics based on the information provided by the designer and the physical connections of joints [62]. Figure 5 presents the Simulink (SimMechanics toolbox) model of the virtual robot.

B. EXPERIMENTAL SETUP

In this research, the experimental setup is composed of a 3-DOF joystick as the master device, and a virtual 6-DOF SCARA type robotic manipulator. This manipulator has a 1-DOF telescopic mast and a 5-DOF robotic manipulator as a slave with an attached working tool for operation at the

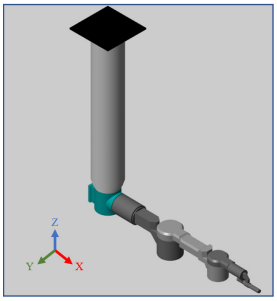

FIGURE 5. Virtual robot Simulink model.

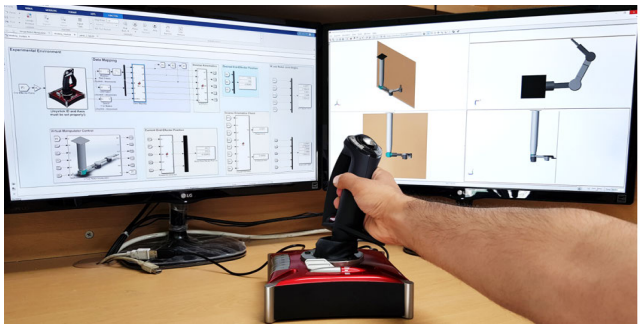

FIGURE 6. Bilateral teleoperation experimental setup.

end-effector in a real hardware. The bilateral teleoperation experimental setup is presented in Figure 6.

The joystick for generating the desired trajectory has a vibration function for haptic feedback. The experimental setup in Figure 6 presents an application of the bilateral teleoperation with the haptic feedback in the field of NPP decommissioning. This is appropriate because the thermal and radioactive environment of the NPP for decommissioning requires humanless or remote automation [4], [63]. The virtual prototype of the manipulator representing the real-time hardware is controlled through the joystick command using the proposed IK and position control algorithm. The Simulink bilateral teleoperation experimental structure is described in Figure 7.

The joystick is connected to the Simulink by USB communication through the Simulink joystick driver block, as presented in Figure 7. Each joystick movement produces a data value in each axis ranging from −1 to 1 (unitless). The data values are then magnified by multiplying them with the constant gain (500 in the current study) depending on the maximum reachable workspace of the manipulator.

The master joystick design is asymmetric with the slave robotic manipulator; hence, the joystick data are used to determine the end-effector's desired position in Cartesian space. Next, the IK, which generates the joint trajectory for the designed position controller, is derived. By using this controller, the SimMechanics model is manipulated and driven. This model represents the actual system hardware and aids in visualizing the system in a parallel monitoring window for a visual feedback.

C. HAPTIC FEEDBACK

The bilateral teleoperation with the haptic feedback allows the user to interact with and feel the environment as well as perform tasks in a remote or inaccessible environment [64]. To provide haptic feedback, the end-effector force should be calculated. The force at the end-effector is determined by using an impedance model representing the environment dynamics [65], [66] given by

$$f_e = B_e\left(\dot{x}_{ef} - \dot{x}_e\right) + K_e(x_{ef} - x_e) \tag{52}$$

where f_e, B_e, K_e, x_{ef}, and x_e are the force exerted by the environment at the end-effector of the robotic manipulator, damping of the environment, stiffness of the environment,

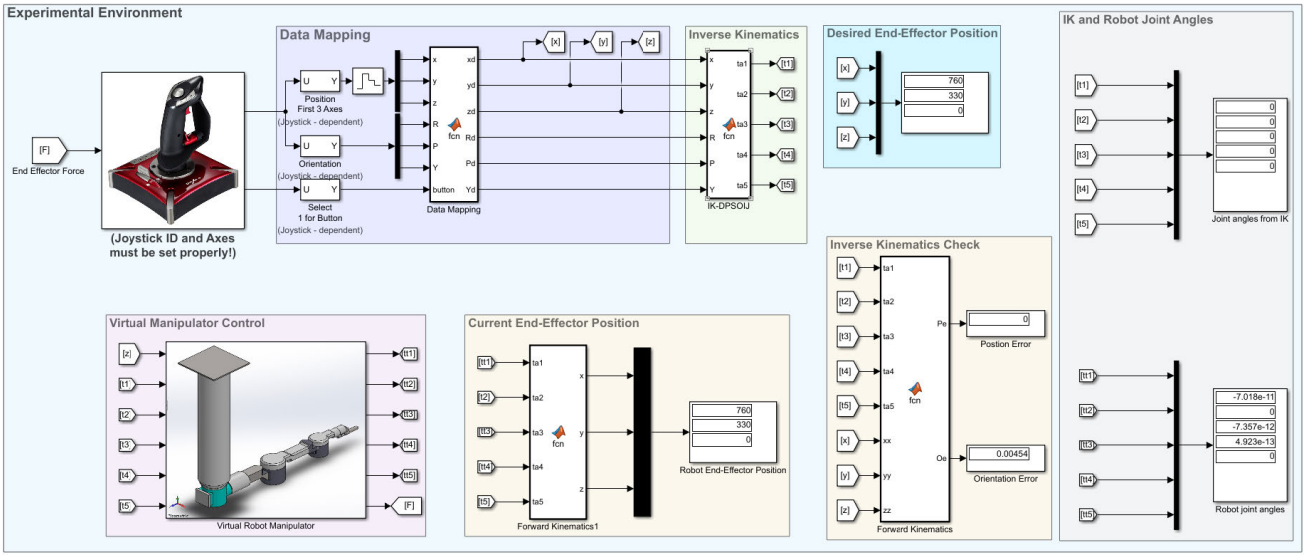

FIGURE 7. Simulink bilateral teleoperation experimental structure.

end-effector position, and environment position, respectively. The joystick used as a master device is PXN- 2119 with a haptic vibrational feedback.

The force calculated using Eq. (52) is then employed to determine the vibration intensity in the master device. The impedance parameters for Eq. (52) are summarized in Table 4.

TABLE 4. Impedance parameters.

Parameters	Value	Unit
B_e	$1e^2$	Kg/sec
K_e	$1e^4$	Kg/sec^2

VI. SIMULATION RESULTS

Before performing an experiment in real time, the IK algorithms and control method are analyzed and validated in the simulations. The simulation results present the initial progress in the system. For both the IK and ISMC control schemes, the simulations and results, which indicate the initial progress in the system, are as follows.

A. IK

Initially, the IK algorithms were simulated without a joystick, and the results were compared and validated based on the allowable error tolerance of position and orientation.

TABLE 5. Desired posture.

No	$P_d\ (x_d, y_d, z_d)$ (mm)	$O_d(R_d, P_d, Y_d)$ (rad)
1	$-990, 642, -1140$	$-2.35, -0.82, -1.74$
2	$-600, 425, -284$	$-1.43, 0.87, -3.03$
3	$-2, 820, -250$	$1.57, 1.57, 0$
4	$1620, 1620, -500$	$-1.57, 0.78, 0.78$

1) IK FOR DESIRED POSTURE

The simulations plan a trajectory for which the algorithms should be implemented. Before establishing the trajectory, the desired position (P_d) and orientation (O_d) of a single Cartesian end-effector are considered and generated randomly, as summarized in Table 5. Initially, the inverse Jacobian was used to solve the IK for the desired end-effector position and orientation listed in Table 5. The results of IK using the inverse Jacobian are summarized in Table 6, where it_{IJ} and T_R are the number of iterations performed and the total runtime consumed by the inverse Jacobian, respectively.

The list in Table 6 indicates that the inverse Jacobian method is not effective in solving the IK because the position error, E_p, is extremely small but the orientation error, E_o, is considerably large, such that it exceeds the allowable error tolerance.

After employing the inverse Jacobian, the traditional PSO and DPSO are implemented; their parameters are summarized in Table 7. The PSO and DPSO convergence curves are

TABLE 6. Inverse Jacobian results for desired postures.

No	E_p (mm)	E_o (rad)	it_{IJ}	T_R
1	.972	1.9	3000	.14
2	.603	1.2	3000	.12
3	.489	2.4	3000	.11
4	.917	1.6	3000	.24

TABLE 7. DPSO parameters.

	PSO / PSO-1	PSO-2
ω	0.9	0.5
ω_d	0.99	0.9
c_1	1.5	0
c_2	1.5	1

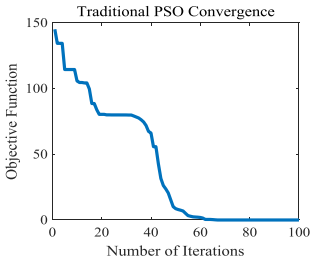

FIGURE 8. PSO convergence.

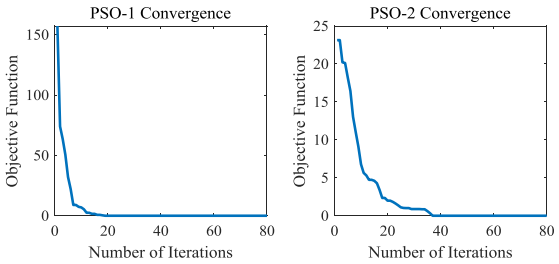

FIGURE 9. DPSO convergence.

TABLE 8. PSO and DPSO comparison.

No	E_{p-PSO}	E_{o-PSO}	E_{p-DPSO}	E_{o-DP}	T_{s-PSO}	T_{s-DPSO}
1	.14	.0013	.45	.005	.242	.325
2	.05	.0023	.88	.0002	.21	.29
3	.04	.0022	.58	.0025	.25	.19
4	.15	.0015	.28	.0005	.28	.22

shown in Figures 8 and 9, respectively. The comparison of the performances of both algorithms is summarized in Table 8. According to Figures 8 and 9, the DPSO is efficient in terms of performing fewer iterations (average iterations: 52) for the desired solution compared with those of the traditional PSO algorithm (average iterations: 66). Initially, the solution of the PSO algorithm was not converging toward the desired result; accordingly, the inertia coefficient was adjusted by multiplying it with a damping coefficient, ω_d, at each iteration. This coefficient is given as follows.

$$\omega = \omega_d \times \omega \tag{53}$$

Another performance comparison between the traditional PSO and DPSO algorithms is summarized in Table 8. Both algorithms are compared based on the position error (PSO position error (E_{p-PSO}) and DPSO position error (E_{p-DPSO})), orientation error (PSO orientation error (E_{o-PSO} in mm) and DPSO orientation error (E_{o-DPSO} in rad)), and total solving time (PSO solution time (T_{C-PSO}) and DPSO solution time (T_{s-DPSO} in s)).

From the list in Table 8, it is observed that the position and orientation errors of both algorithms are within the defined error tolerances, i.e., E_{Tp} and E_{To}, respectively. However, the primary comparison is based on the total simulation time run by both algorithms. For some simple desired postures (i.e., Nos. 3 and 4 in Table 5), the DPSO works well by solving them individually using $\rho_{pso} = 0.25$ and $\sigma_{pso} = 0.75$ in Eq. (25). However, in other cases, the single PSO is more effective, i.e., $\rho_{pso} = \sigma_{pso} = 1$. This means that in some cases, the DPSO does not guarantee the same position by solving the IK within the defined allowable time. In such instances, it is difficult for PSO-2 of the DPSO to attain the desired orientation using the obtained position in PSO-1. This results in solving the IK for position and orientation together; hence, the DPSO changes the position obtained previously. In such cases, the DPSO requires more time to complete the solution to achieve the desired result.

2) IK FOR PLANNED TRAJECTORY

After obtaining the single posture results, the IK algorithms are implemented for a planned trajectory. The trajectory is divided into several points (50 in the current case), and for every point, the IK is performed. The output error graph of position and orientation when implementing the inverse Jacobian, DPSO, and DPSOIJ for the desired end-effector trajectory are shown in Figures 10 and 11, respectively. Where the number of points in the x-axis indicates the number of divisions from the initial point (location/position) to the endpoint of the desired end-effector trajectory.

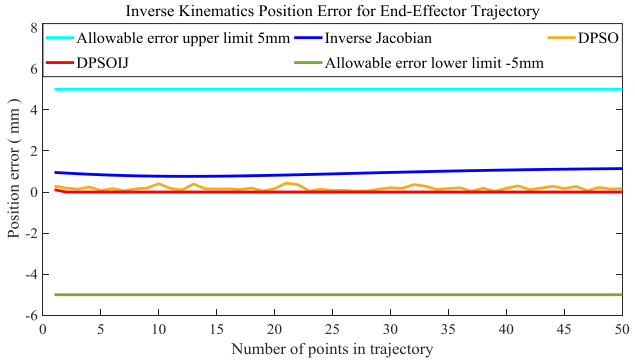

FIGURE 10. IK trajectory position error.

The position errors of all the methods are less than the defined error tolerance, i.e., $E_{Tp} = \pm 5\ mm$; the orientation error of the inverse Jacobian is greater than the defined error tolerance, i.e., $E_{Tp} = \pm 0.0175 rad$ throughout the trajectory.

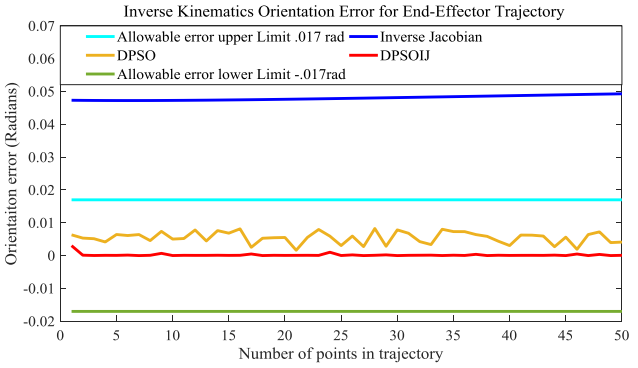

FIGURE 11. IK trajectory orientation error.

It is observed from Figures 10 and 11 that for the inverse Jacobian, the error throughout the trajectory remains practically the same, i.e., $E_p = 1 \pm 0.1mm$ and $E_o = 0.048 + 0.002rad$. This validates the behavior discussed in Section III-C-2, i.e., if the first/starting point of the trajectory has an orientation error, the rest of the trajectory will have practically the same initial error. This means that when using the inverse Jacobian, the first/starting point IK should be precisely calculated. The proposed DPSOIJ exploits this behavior.

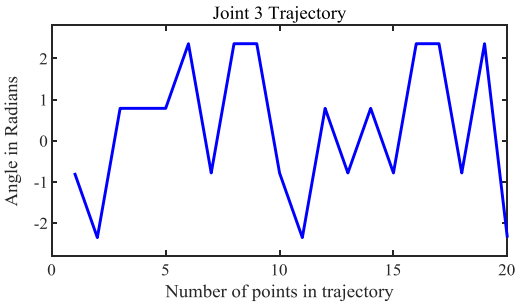

FIGURE 12. DPSO joint 3 trajectory.

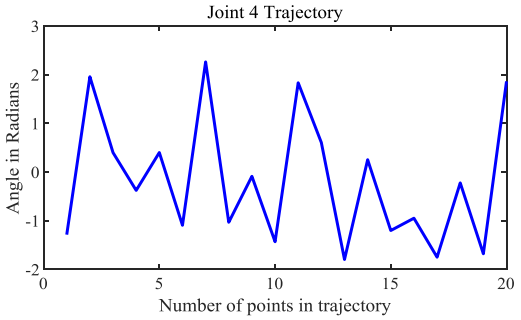

FIGURE 13. DPSO joint 4 trajectory.

In contrast, the trajectories of joints 3 and 4 that are solely generated by the DPSO are presented in Figures 12 and 13, respectively; a considerable angle variation in the generated trajectory is observed. This variation is caused by the heuristic nature of the PSO because it is a heuristic approach.

Consequently, its output may satisfy the given condition, but the process for choosing the best solution is unpredictable

because the solution is randomly generated. The PSO randomly finds a set of angles for one point in the trajectory and selects the best angle. For the second point in the trajectory, the solution is once again started with random numbers resulting in the optimal angle selection for that point based on the cost function (Eqs. (22) and (25)).

At this instance, the angles for the second point in the trajectory considerably differ from the angles obtained for the previous point, even though the two points are extremely near each other. This results in a variating joint trajectory from one point to another, leading to the conclusion that only the DPSO-based IK is not feasible for the control implementation on the manipulator with actual dynamics.

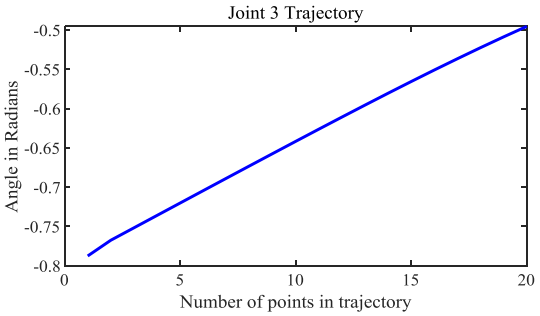

FIGURE 14. DPSOIJ joint 3 trajectory.

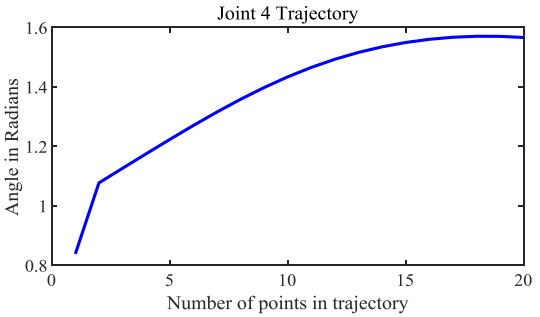

FIGURE 15. DPSOIJ joint 4 trajectory.

Accordingly, in this research, the DPSOIJ is proposed to smooth the trajectory to achieve even motion and control implementation in the robotic manipulator. The joint trajectories of joints 3 and 4 generated by the DPOSIJ are presented in Figures 14 and 15, respectively. When the IK is implemented using the DPSOIJ, the trajectories of both joints (3 and 4) become considerably smooth. Hence, the DPSOIJ is feasible for the control implementation of a planned trajectory because it can generate a smooth joint trajectory with small errors in the end-effector position and orientation.

The aforementioned techniques are also compared based on the total time consumed by each method to solve IK for the desired trajectory divided into 50 points as summarized in Table 9. The fastest solution is the inverse Jacobian. The DPSO is more time-consuming than the inverse Jacobian and DPSOIJ because for each point (position), an average population of 80 is considered with an average of 55 iterations.

TABLE 9. IK simulation time.

Technique	Time (Sec)
Inverse Jacobian	2
DPSO	2.89
DPSOIJ	2.25

The DPSOIJ is less time-consuming than the DPSO but consumes more time than inverse Jacobian because the first point is solved using the DPSO.

B. ISMC IMPLEMENTATION

The ISMC scheme is proposed for the robotic manipulator for decommissioning NPPs. The advantage of ISMC is that it does not require information on system dynamics; that information is compensated by the switching gain, ρ, in Eq. (47). Before the simulation, the controller performance between the ISMC and SMC in the presence of external disturbance was compared to determine the effectiveness of the former; for the controller performance, a second-order system was considered. The controller performance comparison is summarized in Table 10. With the SMC, the system is significantly affected by the external disturbance even with a high switching gain, whereas the effect of disturbance on the system response using the ISMC is minimal.

TABLE 10. Controller performance evaluation.

Controller	Switching Gain	Rising Time	Settling Time $\pm 1\%$	Steady-state error $\pm 1\%$
SMC	700	2.36sec	5.35sec	0
ISMC	50	2.31sec	4.65sec	0

For the robot control implementation, several desired trajectories are introduced. A polynomial trajectory for the prismatic joint is devised, as shown in Figure 16. The ISMC parameters are listed in Table 11; in this case, the parameters are manually tuned by trial and error.

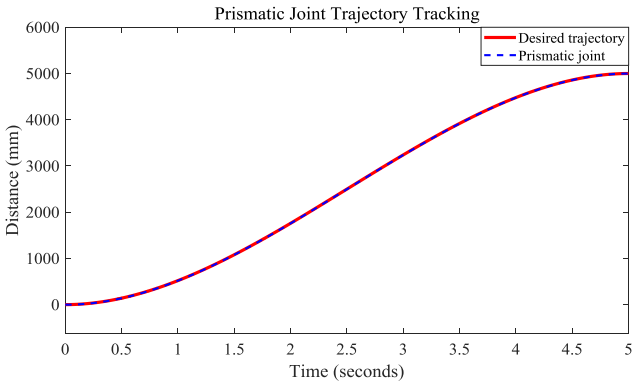

FIGURE 16. Prismatic joint trajectory tracking.

The trajectory tracking error of the prismatic joint is presented in Figure 17. The prismatic joint has a 2 mm steady-state error because of its heavy weight and work

TABLE 11. ISMC parameters.

Controller Parameters	Joints					
	1	2	3	4	5	6
k	150	50	35	25	20	20
c	50	20	20	15	10	10
ρ	300	150	150	120	100	100

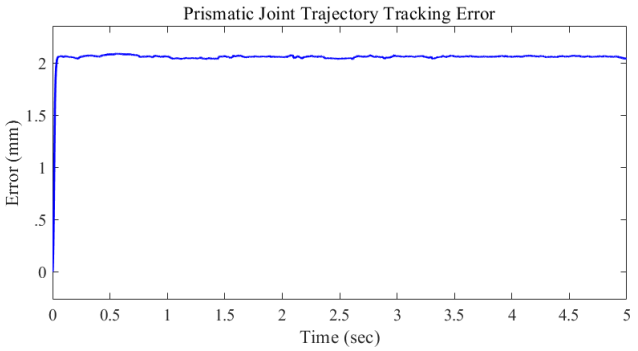

FIGURE 17. Prismatic joint trajectory tracking error.

against gravity. A sine wave with a 0.7854 rad amplitude and 1.25 rad/s frequency is introduced as the desired trajectory for the different joints of the 5-DOF robotic manipulator. The trajectory tracking and tracking error of joints 2 and 4 shown in Figures 18 and 19, respectively, are obtained by observing the controller performances of both joints.

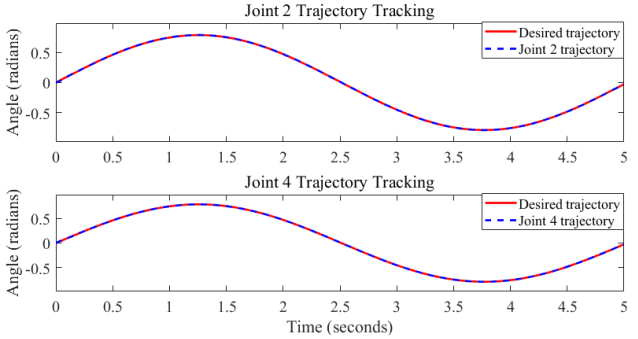

FIGURE 18. Joint 2 and joint 4 trajectory tracking.

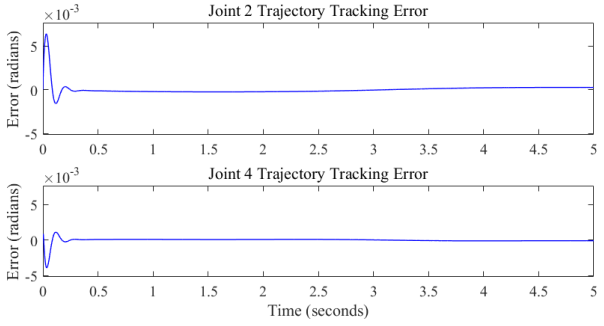

FIGURE 19. Joint 2 and joint 4 trajectory tracking error.

Figures 16–19 demonstrate the improved performance of the ISMC despite the absence of information on the system dynamics. The desired trajectories in the simulations are smooth and continuous. Accordingly, to observe the ISMC performance under a randomly changing trajectory, the controller has to be validated by conducting an experiment in real time on the virtual manipulator using the joystick

VII. EXPERIMENTAL RESULTS

The experimental results of IK and ISMC while teleoperation with the experimental setup presented in Figure 6 and 7 are presented as follows.

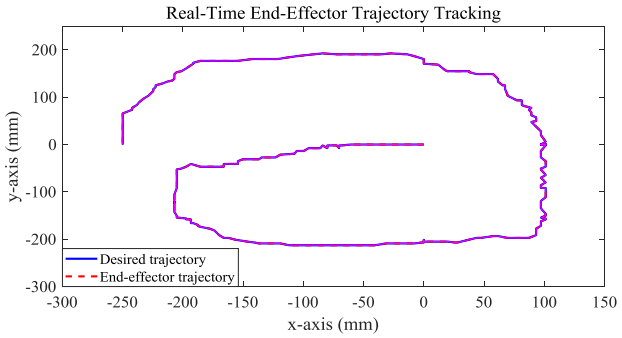

FIGURE 20. End-Effector real-time trajectory tracking.

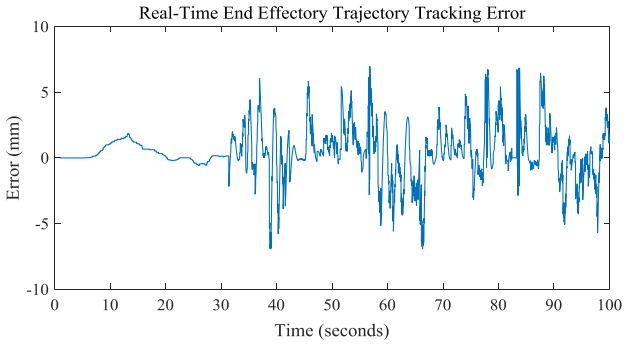

FIGURE 21. End-Effector trajectory tracking error.

A. IK

In the real-time experiment with the joystick, the IK is initially implemented without the virtual manipulator to validate the IK. A random movement of the joystick is performed, generating a random end-effector trajectory (position only; top-view) using the DPSOIJ as presented in Figure 20; the end-effector trajectory tracking error is shown in Figure 21. The DPSO and DPSOIJ are implemented in real-time with the joystick to observe the generated trajectory. Figure 22 presents the generated trajectory of joint 2 using the DPSO and DPSOIJ. The DPSO generates large angle variations as illustrated in Figures 12 and 13. These deviations are caused by the IK and mainly consist of randomly generated solutions for each point or new data; consequently, the joint trajectory is variated. However, in the case of DPSOIJ, the generated trajectory is considerably better and smoother than that produced by the DPSO. Similar to the procedure used in the DPSOIJ, only the first or starting point is calculated using the DPSO; the rest of the trajectory is obtained using the inverse Jacobian to derive a smooth trajectory.

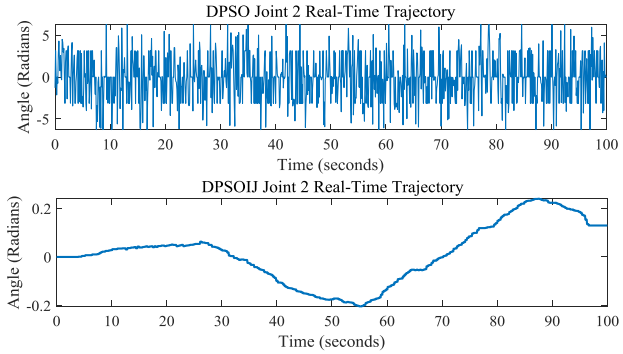

FIGURE 22. Real-time joint 2 trajectory.

The simulation and experimental results of IK indicate that both algorithms (DPSO and DPSOIJ) provide better end-effector trajectory tracking with a small error. They only differ in terms of the nature of the generated joint trajectory, i.e., either variating or smooth (Figure 22). To reduce the calculation time, it should be noted that only the end-effector position is considered in the experiment.

TABLE 12. ISMC parameters for real-time experiment.

Controller Parameters	Joints					
	1	2	3	4	5	6
k	150	100	100	75	40	20
c	50	50	50	40	10	10
ρ	500	150	150	120	100	100

B. ISMC

The ISMC parameters for the real-time experiment are summarized in Table 12. Here, the parameters are tuned manually by trial and error. Numerous experiments are performed on the different types of operations in the NPP dismantling process. For the controller implementation, only the end-effector position is considered, and IK is performed using the inverse Jacobian in which the orientation is assumed parallel to the y-axis. The experimental results include the trajectory tracking of different joints and the trajectory tracking errors of those joints. The trajectory tracking of joint 2 in the real-time teleoperation experiment is presented in Figure 23.

The trajectory tracking error of joint 2 is presented in Figure 24. The trajectory tracking of joint 2 (Figure 23) shows that the ISMC works robustly in real time even without the information on system dynamics. In Figure 24, the blue and red colors represent the actual and expected trajectory tracking errors, respectively. The actual error is greater than the predicted error because it includes the combined errors of the IK and control process.

There are some sudden spikes in the errors, i.e., at approximately 40, 65, and 80 s. These spikes are undesirable; they are not part of the control process but are caused by the fluctuation in joystick data. This problem occurs because the joystick in the current study is used for gaming (PXN2119II); hence, it is inaccurate. The control process assumes that the

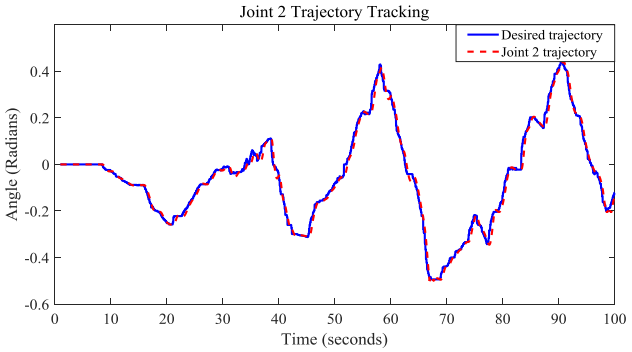

FIGURE 23. Real-time joint 2 trajectory tracking.

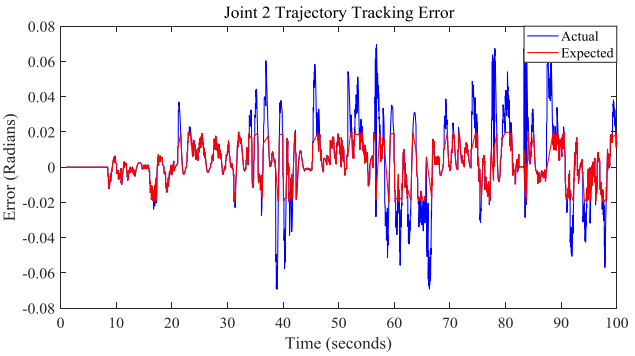

FIGURE 24. Real-time joint 2 trajectory tracking error.

joystick moves fast (this is among the scenarios considered in the simulation) and causes rapid changes in data. This generates a high-velocity trajectory that mismatches the movement of the manipulator, resulting in process delay. Consequently, the error further increases but is tolerable.

The performance of joint 4 can be observed from the trajectory tracking and trajectory tracking error presented in Figures 25 and 26, respectively. Similar to joint 2, joint 4 exhibits acceptable trajectory tracking. However, because of the nature of the joystick (master device), fluctuation errors also occur.

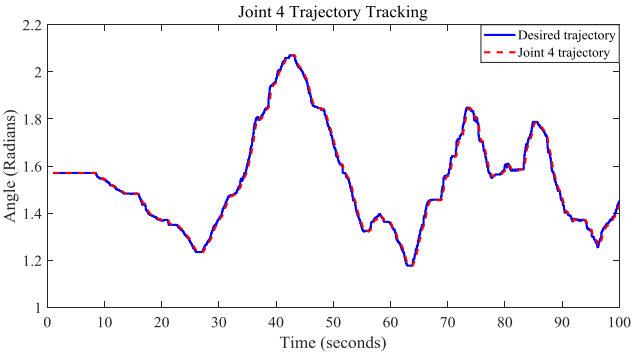

FIGURE 25. Real-time joint 4 trajectory tracking.

The prismatic joint is an important joint to observe in the robotic manipulator because it is the heaviest joint (approximately 735 kg). Hence, it is necessary to monitor the response of the prismatic joint, which changes according the

trajectory generated by the joystick. The trajectory tracking error of the prismatic joint is shown in Figure 27.

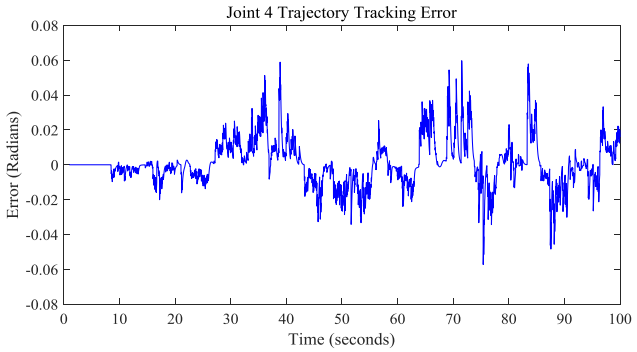

FIGURE 26. Real-time joint 4 trajectory tracking error.

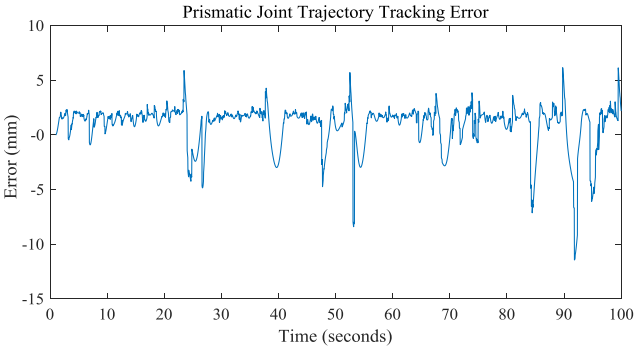

FIGURE 27. Real-time prismatic joint trajectory tracking error.

A close examination of Figure 27 shows that the simulation results of the prismatic joint has a smooth trajectory with a 2 mm average steady-state because of the effect of gravity. The spikes in errors are again caused by the joystick data and randomly generated trajectory. Moreover, it is observed that whenever the generated trajectory is opposite to the direction of gravity, the error (negative error or less than 2 mm) further aggravates. Nevertheless, the controller performance remains satisfactory because the system does not encounter extreme variations.

C. CONTROLLER PERFORMANCE WITH TIME DELAY

This section introduces the variation in the communication time delay over time as given by $t_d = 0.2\sin(t)$. This means that the delay changes from 0–2 s with time when the sampling time of the system is 1 ms. It is observed in the simulations that as a result of this time-varying communication time delay, the operator perceives the end-effector force depending on the delay at that moment, i.e., the response lag between 0 and 2 s. The haptic feedback at the joystick with and without time delay is given in Figure 28, represented by the red line and blue dotted line, respectively.

The communication time delay includes the delay of the data sent from the joystick to the slave and the force feedback received at the joystick. Accordingly, a system with a control process time delay is introduced.

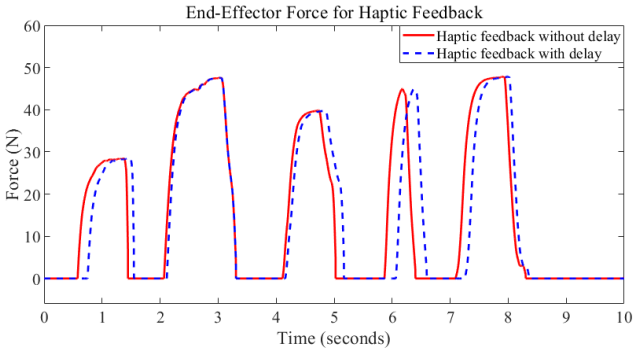

FIGURE 28. Haptic feedback at joystick with variable time delay.

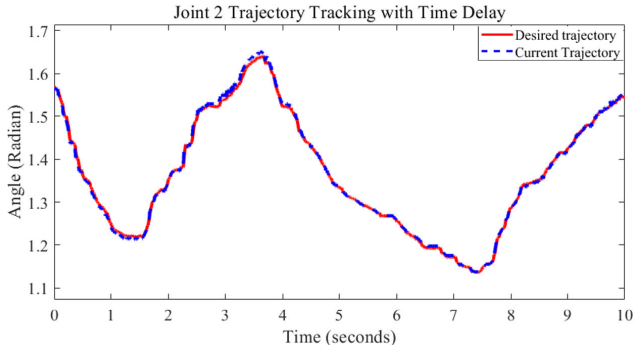

FIGURE 29. Real-time joint 2 ISMC trajectory tracking with time delay.

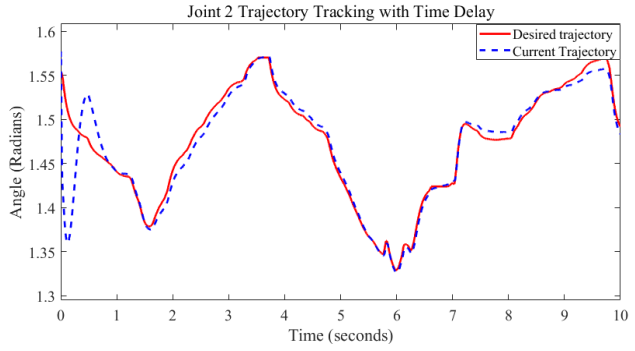

FIGURE 30. Real-time joint 2 SMC trajectory tracking with time delay.

The joystick only generates commands for the slave robot, and the IK and control work on the slave robot area (Figure 4). Therefore, the control process delay aids in validating the ISMC compared with the SMC. The system response under time delay may be observed by transforming the system into a time-varying scheme by introducing time-varying inputs and state delay functions [45], such as $\zeta(t) = 100\sin(t)$ and $d(t) = 100\cos(t)$. The responses of joint 2 with time delay for the ISMC and SMC are shown in Figures 29 and 30, respectively.

The foregoing system transformation has certain effects on system performance. In Figure 29, the response of joint 2 from 2.7 to 4.8 s indicates that the time delay leads to certain tracking errors and chattering. In comparing the ISMC response with that of the SMC in the presence of time delay,

the SMC exhibits an undesirable response (Figure 30) even with high switching gains. The SMC shows significant delays and undershoots with higher errors than the ISMC. In contrast, the overall response of the ISMC is satisfactory.

VIII. CONCLUSION

This paper presents an application of the bilateral teleoperation with a haptic feedback for the NPP dismantling process. The procedure includes the use of a joystick (master device) and virtual robot (slave device). The joystick movement (command) generates the trajectory for the end-effector of the slave in Cartesian space with the control implementation in the joint space. To generate the joint trajectory, the IK was performed. The inverse Jacobian method was initially implemented for the IK for a predefined operation and trajectory. However, the error in the orientation was considerable; therefore, to reduce error, another heuristic approach, the DPSO, was implemented. The DPSO reduced the error by separately solving the position and orientation. The joint trajectory generated by the DPSO-based IK, however, has a considerable angle variation. To solve the error and fluctuation problem simultaneously, another novel IK approach, i.e., the integration of DPSO and inverse Jacobian, as one of the main contributions of this research, was proposed. The DPSOIJ aided in obtaining a smooth joint trajectory with less errors for a planned end-effector trajectory. For precise trajectory tracking, the ISMC was implemented. In the ISMC, the auxiliary sliding surface makes the system insensitive to perturbations (disturbance and unmatched uncertainties), making the controller robust and faster to respond. The ISMC switching gain aids in compensating for the system dynamics; accordingly, the dynamic model information is not required, reducing the control process time delay. The system satisfactorily tracked the trajectory (joystick command). In the future, the authors intend to integrate the non-linear compensator SPO with the ISMC to improve performance. Additional algorithms for solving the communication delay problem will also be introduced. Moreover, a precise joystick or haptic device (such as Geomagic Touch or 3D Touch) with proper data mapping will be utilized to solve the above problem.

REFERENCES

[1] G. Graetz and G. Michaels, "Robots at work," *Rev. Econ. Statist.*, vol. 100, no. 5, pp. 753–768, 2018.

[2] E. Abele, M. Weigold, and S. Rothenbücher, "Modeling and identification of an industrial robot for machining applications," *CIRP Ann.*, vol. 56, no. 1, pp. 387–390, 2007.

[3] W. G. Hao, Y. Y. Leck, and L. C. Hun, "6-DOF PC-based robotic arm (PC-ROBOARM) with efficient trajectory planning and speed control," in *Proc. 4th Int. Conf. Mechatronics (ICOM)*, May 2011, pp. 1–7.

[4] T. Moore, "Robots for nuclear power plants," *IAEA Bull.*, vol. 27, no. 3, pp. 31–38, 1985.

[5] S. Abbasi, K. Kallu, and M. Lee, "Efficient control of a non-linear system using a modified sliding mode control," *Appl. Sci.*, vol. 9, no. 7, p. 1284, Mar. 2019.

[6] F. E. Gelhaus and H. T. Roman, "Robot applications in nuclear power plants," *Prog. Nucl. Energy*, vol. 23, no. 1, pp. 1–33, Jan. 1990.

[7] G. Clement, J. Vertut, A. Cregut, P. Antione, and J. Guittet, "Remote handling and transfer techniques in dismantling strategy," in *Proc. Seminar Remote Handling Nucl. Facilities*, 1984, pp. 556–569.

[8] S. Ma, S. Hirose, and H. Yoshinada, "Development of a hyper-redundant multijoint manipulator for maintenance of nuclear reactors," *Adv. Robot.*, vol. 9, no. 3, pp. 281–300, Jan. 1994.

[9] K. D. Kallu, S. J. Abbasi, H. Khan, J. Wang, and M. C. Lee, "Tele-operated bilateral control of hydraulic manipulator using a robust controller based on the sensorless estimated reaction force," *Appl. Sci.*, vol. 9, no. 10, p. 1995, May 2019.

[10] Z. Chen, F. Huang, C. Yang, and B. Yao, "Adaptive fuzzy backstepping control for stable nonlinear bilateral teleoperation manipulators with enhanced transparency performance," *IEEE Trans. Ind. Electron.*, vol. 67, no. 1, pp. 746–756, Jan. 2020.

[11] N. Mehmood, F. Ijaz, Z. Murtaza, and S. I. Ali Shah, "Analysis of end-effector position and orientation for 2P-3R planer pneumatic robotic arm," in *Proc. Int. Conf. Robot. Emerg. Allied Technol. Eng. (iCREATE)*, Apr. 2014, pp. 47–50.

[12] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, "Motion control," in *Robotics: Modelling, Planning and Control*. London, U.K.: Springer-Verlag, 2010, pp. 303–360.

[13] P.-F. Lin, M.-B. Huang, and H.-P. Huang, "Analytical solution for inverse kinematics using dual quaternions," *IEEE Access*, vol. 7, pp. 166190–166202, 2019.

[14] K. S. Fu, R. Gonzalez, and C. G. Lee, "Robot arm kinematics," in *Robotics: Control Sensing. Vis*. New York, NY, USA: McGraw-Hill, 1987, pp. 12–76.

[15] S. B. Niku, "Kinematics of robot: Position analysis," in *Introduction to Robotics: Analysis, Systems, Applications*. Upper Saddle River, NJ, USA: Prentice-Hall, 2010, pp. 33–113.

[16] S. O. Park, J. G. Yoon, M. G. Jung, and M. C. Lee, "Robot manipulator arm inverse kinematics analysis by Jacobian," in *Proc. ICAROB Annu. Conf.*, 2018, p. 45.

[17] G. K. Singh and J. Claassens, "An analytical solution for the inverse kinematics of a redundant 7DoF manipulator with link offsets," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2010, pp. 2976–2982.

[18] W. Wolovich and H. Elliott, "A computational technique for inverse kinematics," in *Proc. 23rd IEEE Conf. Decis. Control*, Dec. 1984, pp. 1359–1363.

[19] B. Siciliano, "Kinematic control of redundant robot manipulators: A tutorial," *J. Intell. Robot. Syst.*, vol. 3, no. 3, pp. 201–212, 1990.

[20] T. P. Singh, P. Suresh, and S. Chandra, "Forward and inverse kinematic analysis of robotic manipulators," *Int. Res. Hournal Eng. Technol.*, vol. 4, no. 2, pp. 1459–1469, 2017.

[21] X. Zhang and C. A. Nelson, "Multiple-criteria kinematic optimization for the design of spherical serial mechanisms using genetic algorithms," *J. Mech. Des.*, vol. 133, no. 1, Jan. 2011, Art. no. 011005.

[22] T. Çavdar, M. Mohammad, and R. A. Milani, "A new heuristic approach for inverse kinematics of robot arms," *Adv. Sci. Lett.*, vol. 19, no. 1, pp. 329–333, Jan. 2013.

[23] N. Rokbani and A. M. Alimi, "Inverse kinematics using particle swarm optimization, a statistical analysis," *Procedia Eng.*, vol. 64, pp. 1602–1611, Jan. 2013.

[24] J. de Lima Silveira Junior, R. C. de Oliveira Jesus, L. Molina, E. A. N. Carvalho, and E. Oliveira Freire, "FRPSO: Inverse kinematics using fully resampled particle swarm optimization," in *Proc. Latin Amer. Robotic Symp., Brazilian Symp. Robot. (SBR) Workshop Robot. Educ. (WRE)*, Nov. 2018, pp. 402–407.

[25] T. J. Collinsm and W.-M. Shen, "Particle swarm optimization for high-DOF inverse kinematics," in *Proc. 3rd Int. Conf. Control, Autom. Robot. (ICCAR)*, Apr. 2017, pp. 1–6.

[26] M. A. Adly and S. K. Abd-El-Hafiz, "Inverse kinematics using single- and multi-objective particle swarm optimization," in *Proc. 28th Int. Conf. Microelectron. (ICM)*, Dec. 2016, pp. 269–272.

[27] R. Falconi, R. Grandi, and C. Melchiorri, "Inverse kinematics of serial manipulators in cluttered environments using a new paradigm of particle swarm optimization," in *Proc. IFAC*, 2014, vol. 47, no. 3, pp. 8475–8480.

[28] H.-C. Huang, C.-P. Chen, and P.-R. Wang, "Particle swarm optimization for solving the inverse kinematics of 7-DOF robotic manipulators," in *Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC)*, Oct. 2012, pp. 3105–3110.

[29] Y. Du and Y. Wu, "Application of IPSO algorithm to inverse kinematics solution of reconfigurable modular robots," in *Proc. Int. Conf. Mech. Sci., Electric Eng. Comput. (MEC)*, Aug. 2011, pp. 1313–1316.

[30] H. Khan, H. H. Kim, S. J. Abbasi, and M. C. Lee, "Real-time inverse kinematics using dual particle swarm optimization DPSO of 6-DOF robot for nuclear plant dismantling," in *Proc. IFAC*, Berlin, Germany, 2020, pp. 1–6.
[31] M. A. Johnson and M. H. Moradi, "PID control technology," in *PID Control*. London, U.K.: Springer, 2005, pp. 1–46.
[32] E. M. Jafarov, M. N. A. Parlakci, and Y. Istefanopulos, "A new variable structure PID-controller design for robot manipulators," *IEEE Trans. Control Syst. Technol.*, vol. 13, no. 1, pp. 122–130, Jan. 2005.
[33] J. Z. Shi, "A fractional order general type-2 fuzzy PID controller design algorithm," *IEEE Access*, vol. 8, pp. 52151–52172, 2020.
[34] J. Tang, F. Huang, Z. Chen, T. Wang, J. Gu, and S. Zhu, "Disturbance-observer-based sliding mode control design for nonlinear bilateral teleoperation system with four-channel architecture," *IEEE Access*, vol. 7, pp. 72672–72683, 2019.
[35] K. D. Young and U. Ozguner, "Sliding mode: Control engineering in practice," in *Proc. Amer. Control Conf.*, vol. 1, 1999, pp. 150–162.
[36] X.-G. Yan, S. K. Spurgeon, and C. Edwards, "Introduction," in *Variable Structure Control of Complex Systems*. Cham, Switzerland: Springer, 2017, pp. 1–25.
[37] Z. Chen, F. Huang, W. Chen, J. Zhang, W. Sun, J. Chen, J. Gu, and S. Zhu, "RBFNN-based adaptive sliding mode control design for delayed nonlinear multilateral telerobotic system with cooperative manipulation," *IEEE Trans. Ind. Informat.*, vol. 16, no. 2, pp. 1236–1247, Feb. 2020.
[38] X. Liu, W. Jiang, and X.-C. Dong, "Nonlinear adaptive control for dynamic and dead-zone uncertainties in robotic systems," *Int. J. Control, Autom. Syst.*, vol. 15, no. 2, pp. 875–882, Apr. 2017.
[39] J. T. Moura, H. Elmali, and N. Olgac, "Sliding mode control with sliding perturbation observer," *J. Dyn. Syst., Meas., Control*, vol. 119, no. 4, pp. 657–665, Dec. 1997.
[40] H. Elmali and N. Olgac, "Implementation of sliding mode control with perturbation estimation (SMCPE)," *IEEE Trans. Control Syst. Technol.*, vol. 4, no. 1, pp. 79–85, Jan. 1996.
[41] K. D. Kallu, S. J. Abbasi, H. Khan, J. Wang, and M. C. Lee, "Implementation of a TSMCSPO controller on a 3-DOF hydraulic manipulator for position tracking and sensor-less force estimation," *IEEE Access*, vol. 7, pp. 177035–177047, 2019.
[42] H. Khan, S. J. Abbasi, K. Dad Kallu, and M. C. Lee, "Robust control design of 6-DOF robot for nuclear power plant dismantling," in *Proc. Int. Conf. Robot. Autom. Ind. (ICRAI)*, Oct. 2019, pp. 1–7.
[43] Y. Pan, C. Yang, L. Pan, and H. Yu, "Integral sliding mode control: Performance, modification, and improvement," *IEEE Trans. Ind. Informat.*, vol. 14, no. 7, pp. 3087–3096, Jul. 2018.
[44] Y. Pan, Y. H. Joo, and H. Yu, "Discussions on smooth modifications of integral sliding mode control," *Int. J. Control, Autom. Syst.*, vol. 16, no. 2, pp. 586–593, Apr. 2018.
[45] D. B. Salem, W. Saad, A. Sellami, and G. Garcia, "Integral sliding mode control for systems with time-varying input and state delays," in *Proc. Int. Conf. Eng. MIS (ICEMIS)*, May 2017, pp. 1–5.
[46] Z. Chen, F. Huang, W. Sun, J. Gu, and B. Yao, "RBF-neural-network-based adaptive robust control for nonlinear bilateral teleoperation manipulators with uncertainty and time delay," *IEEE/ASME Trans. Mechatronics*, vol. 25, no. 2, pp. 906–918, Apr. 2020.
[47] F. Huang, W. Zhang, Z. Chen, J. Tang, W. Song, and S. Zhu, "RBFNN-based adaptive sliding mode control design for nonlinear bilateral teleoperation system under time-varying delays," *IEEE Access*, vol. 7, pp. 11905–11912, 2019.
[48] H. Caballero-Barragán, L. P. Osuna-Ibarra, A. G. Loukianov, and F. Plestan, "Robust control for perturbed linear systems with time-varying delay via sliding mode control," in *Proc. 15th Int. Workshop Variable Struct. Syst. (VSS)*, Jul. 2018, pp. 7–12.
[49] Y. Shtessel, C. Edwards, L. Fridman, and A. Levant, "Introduction," in *Sliding Mode Control and Observation*. New York, NY, USA: Springer, 2014, pp. 1–42.
[50] H. Yi and Q. Zhang, "An optimal fuzzy control method for nonlinear time-delayed batch processes," *IEEE Access*, vol. 8, pp. 42608–42618, 2020.
[51] M. W. Spong and M. Vidyasagar, "Velocity kinematics: The manipulator Jacobian," in *Robot Dynamics and Control*. Hoboken, NJ, USA: Wiley, 2008, pp. 112–128.
[52] S. Park and M. C. Lee, "7DOFs robot numerical approach method with Jacobian," in *Proc. Int. Conf. Inf. Commun. Technol. Robot. (ICT-ROBOT)*, Sep. 2018, pp. 1–4.
[53] K. M. Lynch and F. C. Park, "Inverse kinematics" in *Modern Robotics: Mechanics, Planning and Control*. Cambridge, U.K.: Cambridge Univ. Press, 2017, pp. 219–244.
[54] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proc. 6th Int. Symp. Micro Mach. Hum. Sci. (MHS)*, 1995, pp. 39–43.
[55] R. Poli, J. Kennedy, and T. Blackwell, "Particle swarm optimization," *Swarm Intell.*, vol. 1, no. 1, pp. 33–57, Jun. 2007.
[56] İ. Eker, "Second-order sliding mode control with experimental application," *ISA Trans.*, vol. 49, no. 3, pp. 394–405, Jul. 2010.
[57] X. Zhou, W. Wang, Z. Liu, C. Liang, and C. Lai, "Impact angle constrained three-dimensional integrated guidance and control based on fractional integral terminal sliding mode control," *IEEE Access*, vol. 7, pp. 126857–126870, 2019.
[58] C. P. Vo, X. D. To, and K. K. Ahn, "A novel adaptive gain integral terminal sliding mode control scheme of a pneumatic artificial muscle system with time-delay estimation," *IEEE Access*, vol. 7, pp. 141133–141143, 2019.
[59] V. Nath and R. Mitra, "Swing-up and control of rotary inverted pendulum using pole placement with integrator," in *Proc. Recent Adv. Eng. Comput. Sci. (RAECS)*, Mar. 2014, pp. 1–5.
[60] K. J. Åström and K. Furuta, "Swinging up a pendulum by energy control," *Automatica*, vol. 36, no. 2, pp. 287–295, Feb. 2000.
[61] Y. Shaoqiang, L. Zhong, and L. Xingshan, "Modeling and simulation of robot based on MATLAB/SimMechanics," in *Proc. 27th Chin. Control Conf.*, Jul. 2008, pp. 161–165.
[62] A. Kathpal and A. Singla, "SimMechanics based modeling, simulation and real-time control of rotary inverted pendulum," in *Proc. 11th Int. Conf. Intell. Syst. Control (ISCO)*, Jan. 2017, pp. 166–172.
[63] H. Shin, S. H. Jung, Y. R. Choi, and C. Kim, "Development of a shared remote control robot for aerial work in nuclear power plants," *Nucl. Eng. Technol.*, vol. 50, no. 4, pp. 613–618, May 2018.
[64] X. Xu, B. Cizmeci, C. Schuwerk, and E. Steinbach, "Model-mediated teleoperation: Toward stable and transparent teleoperation systems," *IEEE Access*, vol. 4, pp. 425–449, 2016.
[65] L. Roveda, "Adaptive interaction controller for compliant robot base applications," *IEEE Access*, vol. 7, pp. 6553–6561, 2019.
[66] F. Zeng, J. Xiao, and H. Liu, "Force/torque sensorless compliant control strategy for assembly tasks using a 6-DOF collaborative robot," *IEEE Access*, vol. 7, pp. 108795–108805, 2019.

HAMZA KHAN received the B.S. degree in mechatronics engineering from Air University, Islamabad, Pakistan, in 2018. He is currently pursuing the M.S. degree in mechanical engineering with Pusan National University, Busan, South Korea. His research interests include non-linear control, robot manipulators, and system identification.

SAAD JAMSHED ABBASI received the B.S. degree in mechatronics engineering from Air University, Islamabad, Pakistan, in 2015, and the M.S. degree in mechanical engineering from Pusan National University, Busan, South Korea, in 2018, where he is currently pursuing the Ph.D. degree in mechanical engineering. His research interests include non-linear control, robot manipulators, and system identification.

MIN CHEOL LEE (Member, IEEE) received the Ph.D. degree in applied physics from the University of Tsukuba, Tsukuba, Japan, in 1991. He was a Visiting Professor with North Carolina State University from August 2000 to 2001, and with Perdue University from 2009 to 2010. Since 1991, he has been a Professor with the School of Mechanical Engineering, Pusan National University, South Korea. His research interests include intelligent robot control, autonomous mobile robot, medical robot, signal processing to identify a systems, robust control of a systems, sensor application, and mechatronics.